Journal of Chemical Technology and Metallurgy, 50, 6, 2015, 595-599

NEUROSTRUCTURAL MODELLING AND PREDICTION OF HOT-ROLLED PRODUCTION DEFECTS BY CASTING PARAMETERS

Anatoly Pogodaev, Pavel Saraev

Lipetsk State Technical University
Moskovskaya st., 30, 398036,
Lipetsk, Russian Federation
E-mail: psaraev@yandex.ru

Received 19 February 2015
Accepted 26 June 2015

ABSTRACT

The aim of the work is to predict defects of hot-rolled production based on neurostructural modelling. Prediction is made by parameters of ladle treatment and continuous casting. Class of neurostructural modelling and algorithm for models construction is used to solve the problem.

Keywords: defects prediction, hot-rolled production, continuous casting, neurostructural modelling, model construction.

INTRODUCTION

The problem of defects detection for hot-rolled production by the set of technological parameters of ladle treatment and continuous casting was investigated. Description of input data is given in Table 1. Observable rolling defects are:

- longitudinal cracks;
- transverse cracks;
- friction sockets and net cracks;
- bubbles;
- slag inclusions;
- skins;
- expanded cracks.

The number of rows with defect skins and expanded cracks are about 7,5 % of all data. The number of rows with casting and rolling defects are 0.3 % of all data. So, the problem of influence of casting parameters on rolling defects was stated. We analyzed the influence of the following factor groups:

- chemical composition;
- parameters of ladle treatment;
- parameters of a steel ladle;
- tundish;
- crystallizer pan;
- casting temperature;
- speed mode;
- aftercooling parameters;
- parameters of soft compression and configuring stream;
- elongation parameters.

EXPERIMENTAL

We processed the input data and deleted the factors with equal or incomplete values. Then we used the correlation analysis and principal component analyses. As a result we selected the factors with significant influence on rolling defects. At final stage we got input table with 38 factors and their linear combinations.

For mathematical modelling we coded rolling production by 0 and production with any defect as 1. So, we have got a problem of classification. We normalized the input data into the range [0; 1]. We compared the model output y with a threshold T. We realized the classification function

$$c(y) = \begin{cases} 1, & \text{if} \quad y \geq T; \\ 0, & \text{otherwise.} \end{cases}.$$

One of the most popular approaches for data modelling is neural networks modelling [1, 2]. Its extension is neurostructural approach. Neurostructural modelling

Table 1. Description of input data.

Parameter	Value
Number of factors	227
Number of rows	16628
Rolling defects	1255
Number of casts	71

(NSM) is generalization of neural modelling for the wide mathematical models classes including feed-forward neural networks, some other neural architectures, fuzzy Takagi-Sugeno models and hybrid neuro-fuzzy models ANFIS (Adaptive Neuro-Fuzzy Implication System) [3].

The structure of neurostructural model with *m* layers is superpositional with linear and nonlinear weights:

$$y = W^{(m)}\sigma^{(m)}\left(\ldots\left(W^{(2)}\sigma^{(2)}\left(W^{(1)}x\right)\right)\ldots\right),$$

where $W^{(i)}$ is the matrix of *i*-th layer weights; $\sigma^{(i)}$ is the vector function of vector argument, constructed from transfer functions of neuron-like elements of *i*-th layer; $i = 1, \ldots, m$, x and y are input and output vectors correspondently. Instead of artificial neuron we use neuron-like element (NLE). NLE generalizes usual neuron by using any continuous transfer function from a class Ω. So, one NSM can contain different functions such as the most popular sigmoid function

$$\sigma(net) = \frac{1}{1+e^{-net}},$$

where

net is the NLE activation level, or Gaussian function

$$\sigma(net) = e^{-\frac{(net-m)^2}{2s^2}},$$

where *m* and *s* are some fixed parameters, or any trigonometric function, for instance.

Taking into account the special structure of NSM it allows developing and applying common algorithmic base for construction and training algorithms. The main problems of NSM building are structure and parameters identifications.

One of the approaches for structure identification is pruning technique [2]. Unfortunately, it has a lot of disadvantages. The most perspective approach for NSM structure building is constructive [2]. Its idea is sequential increasing of number of NLEs and connections between them. This is the way from simple to more complex mathematical models.

We suggested algorithm for NSM construction which guarantees error decreasing during addition of new neuron-like elements [3]. The algorithm guarantees that training error monotone increases while the structure of NSM increases too. Let us consider it in details.

Initial parameter of algorithm is the set of admissible transfer functions Ω. It necessarily contains the linear function $\sigma(net) = net$.

Dimensions of input and output layers are known.

Algorithm of NSM construction is the following.

Step 1. Choose a type of NSM. It may be feed-forward neural network, neural network with RBF functions, probabilistic neural network, Takagi-Sugeno model or any other NSM structure.

Step 2. Set initial structure of NSM with one hidden layer which contains only one NLE.

Step 3. Choose maximum structure of NSM that is maximum number of hidden layers and NLEs in each layer.

Step 4. Train initial structure by any training algorithm. Save the structure with all weights coded as S_0.

Step 5. Add NLE at last hidden layer and set 0 to weight connecting it and output NLE.

Step 6. Retrain NSM for each transfer functions from the set Ω. After all we choose the transfer function providing minimal training error. Save the structure with all weights coded as *S1*. Restore the structure coded as *S0*.

Step 7. Add a new hidden layer before an output layer with the only NLE. Set 1 to weight connecting it and output NLE.

Step 8. Retrain NSM for each transfer functions from the set Ω. After all we choose the transfer function providing minimal training error. Save the structure with all weights coded as *S2*.

Step 9. Choose the best structure from *S1* and *S2* by minimal value of error. Save the chosen structure and weights as *S0*.

Step 10. Finish the construction process if the final structure is achieved. Otherwise go to step 5.

The key idea of the algorithm is setting initial weight values for added NLEs. It allows one to guarantee that new training results starting from structure *S0* will not be worse than before retraining.

Obtained NSM can have different transfer functions at each NLE. This is the advantage that helps to fill the

data with the minimal structure. The algorithm can be complemented by control of decreasing of training error at each step. If the decreasing of training error is sufficiently small (less than small positive real number ε) than the NSM construction can be terminated.

Training NSM is a parameters identification problem. Often it is a nonlinear least squares problem. We have to determine optimal weights *w* which minimize the function

$$Q(w) = \sum_{k=1}^{K} \sum_{q=1}^{Q} \left(y_q(w, \tilde{x}^{(k)}) - \tilde{y}_q^{(k)} \right)^2 ,$$

where $\tilde{x}^{(k)} \in R^P$ is the input vector, $\tilde{y}^{(k)} \in R^Q$ is the output vector, $y_q(w, x^{(k)})$ is the *q*-th model output calculated for the vector $\tilde{x}^{(k)}$ from training set, $\tilde{y}_q^{(k)}$ is the *q*-th element of the training output from *k*-th row. For training NSM we used the algorithm based on interval approach [4, 5]. It guarantees the global minimum of the training problem for multi-extremes function [6].

The results of global optimization may be guaranteed by the methods based on the interval analysis. The interval is connected subset of the set of real numbers defined by a pair of real numbers

$$[x] = \left[\underline{x}, \overline{x}\right] = \left\{x \in R : \underline{x} \le x \le \overline{x}\right\}$$

where $\underline{x}$ and $\overline{x}$ are, respectively, the lower and upper boundaries of the interval. Arithmetic operations can be extended for intervals and interval vectors and matrices [4, 5]. The interval functions are the images of ordinary functions with intervals as arguments:

$$f([x]) = \{f(x) : x \in [x]\}$$

Interval arithmetic operations and interval functions have very important property. They are monotone in inclusion:

$$[x] \subset [x'], [y] \subset [y'] \Rightarrow [x] * [y] \subset [x'] * [y'],$$

for any arithmetic operator *, and

$$[x] \subset [x'] \Rightarrow f([x]) \subset f([x']).$$

An interval algorithm for global optimization is essentially use monotone properties.

The main idea of the interval methods of global optimization lies in successive decomposition of the initial box [*w*] into sub-boxes and estimation on them of the image of the objective function. Convergence is ensured by the monotonicity of the interval functions. Calculations are stopped when the box width becomes smaller than the prescribed precision.

Let $f(w)$, $w \in R^s$ is the objective (minimized) function, $[w] \in IR^s$ is the initial interval vector, where IR^s is the set of *s*-dimension interval vectors, $\varepsilon > 0$ is the minimal permissible width of the box.

The basic interval algorithm GlobOpt for global optimization is the following [5]:

Step 1. Initialize: $[p] := [w]$.

Step 2. Estimate of the minimum: $f^* = [f]([p])$, where $[f]$ is the inclusion function for *f*.

Step 3. Initialize of the list: $L := \{([p], f^*)\}$.

Step 4. As long as the width of the inclusion function

$$wid([f]([p])) \ge \varepsilon ,$$

where *wid* is the width of interval vector, repeat:

(a) Select the component *l* for which the box [*p*] has the greatest width: $l = \arg\max_{i=1}^{n} wid([p_i])$.

(b) Bisect [*p*] by the *l*th coordinate into [*p'*] and [*p''*].

(c) Calculate

$$f' = [f]([p']), \quad f'' = [f]([p'']) .$$

(d) Remove the pair $([p], f*)$ from the list *L*.

(e) Add the pairs $([p'], f')$ and $([p''], f'')$ to the list *L* in increase of the second field.

(f) Notate by $([p], f*)$ of the first (leading) record in the list *L*.

The result of the algorithm work is the interval vector [*p*] containing the global solution of the problem. Some aspects of application of interval methods to neural networks training were considered in works [7 - 9].

The basic interval algorithm for global optimization cannot be used in solving practical problems because with growing dimensionality the effect of bisection becomes less and less appreciable [6]. In interval algorithms it is important to develop effective procedures of elimination of boxes which do not contain any solution of the problem. Such procedures are called contractor operators [4]. We suggested a number of contractor operators which take into account special superpositional structure of NSM and weights separation into linear and non-linear weights. Analysis and discussion of contractor operators for NSM training can be found in detail at [10].

For problem solving we developed software for neurostructural modelling. It was developed in C++ in

Table 2. Classification results for different models.

Model No.	Number of NLEs	Accuracy of classification		
		for training set	for testing set	for primary set
1	1 (Gaussian)	0.82	0.71	0.81
2	1 (sigmoid)	0.82	0.73	0.81
3	2 (two Gaussian functions)	0.84	0.76	0.84
4	2 (two sigmoid functions)	0.84	0.77	0.85
5	3 (different functions)	0.84	0.75	0.82
6	4 (different functions)	0.86	0.78	0.80
7	5 (different functions)	0.86	0.78	0.80
8	6 (different functions)	0.87	0.80	0.86

CodeGear 2009 C++ Builder environment. It can run under operating system Windows XP and later versions. The software realizes algorithms of NSM building based on suggested constructive algorithm and interval algorithm for global optimization. Its main functions are:

- loading the training set of input and output data from the keyboard or from files of format Microsoft Excel and CSV (Comma Separated Values);
- primary preprocessing including normalization of input data into the range [-1; 1] or into the range [0; 1];
- setting the type and structure of NSM;
- possibility of use of usual and weighted function for training;
- setting admissible transfer functions Ω;
- setting construction and training parameters;
- control of progress of NSM building;
- possibility of stopping training procedure if satisfactory training error occurs;
- loading and saving structure, weights and general results of NSM work at the XML-format file;
- computation of the threshold *T* for classification problems;
- calculation of outputs based on the NSM and saving of results to text file.

RESULTS AND DISCUSSION

For better results we trained NSM with weighted function. The weight *w=12.25* (ratio of defects to non-defects production) was used for rows containing defects. So we took into account heterogeneous character of input information.

For estimation of adequacy we divided the given data set in two sets: training and testing. Training set had 80 % of data. We did not use other 20 % of data for training. Estimation for accuracy for testing set was 80 %.

Table 2 shows the results of classification for some NSMs. For first four models we did not use the constructive algorithm. We show results for more full comparative analysis.

We see that training errors are smaller for more large NSMs. Also we see that the decreasing of the error for training set does not guarantee the decreasing of error for testing set. We have chosen the 8-th model because it shows the best results.

So, we built the following NSM based on the suggested construction algorithm and interval training algorithm (model 8 from Table 2):

$$y = w_0 + \sum_{i=1}^{6} w_i \sigma_i \left(w_{i0} + \sum_{j=1}^{38} w_{ij} x_j \right)$$

Table 3. Results of hot-rolled defects classification.

Real class	Model class		Accuracy
	0	1	
0	13658	1688	0.89
1	681	574	0.46
Summary			0.86

where σ_i are transfer functions

$$\sigma_1 = \sigma_2 = \sigma_5 = \sigma_6 = \frac{1}{1+e^{-net}}; \sigma_3 = \sigma_4 = e^{-\frac{net^2}{2}}.$$

Number of weights for the NSM is 241. The error for training set (sum of squared errors, SSE) is $Q(w) \approx 5548.09$.

We determined the threshold T = 0.7 by the criterion

$$T = \arg\min \sum_{i=1}^{16628} w_i \left| c(y_i) - \tilde{y}_i \right|$$

where w_i and $c(y_i)$ are weight and class for the *i*-th example correspondently.

Results of defects classification of the best NSM for testing set are given in Table 3. It shows that approximately 50 % of production with defects can be detected by obtained mathematical model.

Adequacy of the NSM model for real data not from the training and testing sets is approximately 72 %. It is less than error on training and testing sets but suitable for real metallurgical production.

CONCLUSIONS

We obtained neurostructural model for hot-rolled defects prediction based only on parameters of ladle treatment and continuous casting. It was determined by the neurostructural modelling, constructive and interval training algorithms suggested by the authors in earlier works. The model gives appropriate results for real data. It lets one choose optimal trajectory for product processing.

Acknowledgements

The research is supported by the Ministry of Education and Science of the Russian Federation in the bounds of the list of scientific and research work of main part of the government job, project number 970.

REFERENCES

1. S. Haykin, Neural Networks and Learning Machines (3rd Ed.), Prentice Hall, 2009, p. 936.
2. S. Osovskii, Neural Information Processing Networks, Moscow, Finansy i Statistika, 2002, p. 344, (in Russian).
3. P.V. Saraev, Development of Complex Systems Neural Networks Modelling based on Neurostructural Approach, Vesti VUZov Chernozemja, 2, 28, 2012, 30-35, (in Russian).
4. L. Jaulin, M. Kieffer, O. Didrit, E. Walter. Applied Interval Analysis, London, Springer, 2001, p. 379.
5. S.P. Shary, Finite Interval Analysis. Novosibirsk: Institute of Computer Technologies of RAS, 2013, p. 606, http://www.nsc.ru/interval/Library/InteBooks/SharyBook.pdf
6. E. Hansen, G.W. Walster, Global Optimization Using Interval Analysis, New York, Marcel Dekker, 2003, p. 492.
7. E. de Weerdt, Q.P. Chu, J.A. Mulder, Neural Network Global Optimization Using Interval Analysis, in Proc. 27th Benelux Meeting Syst. Control, Heeze, The Netherlands, Netherlands Organisat. Sci. Res., 2008, p. 162.
8. E. de Weerdt, Q.P. Chu, J.A. Mulderm, Neural Network Output Optimization Using Interval Analysis, IEEE Trans. Neural Networks, 2009, 20, 4, 2009, 638-653.
9. F. Rossi, B. Conan-Guez, Multilayer Perceptron on Interval Data, in Proc. Classificat., Clustering, and Data Anal, (IFCS 2002), Poland, Cracow, Springer, 2002, p. 427-434.
10. P.V. Saraev, Numerical Methods of Interval Analysis in Learning Neural Network,/ Automation and Remote Control, 73, 11, 2012, 1865-1876.